REPOI

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1994	Professional Paper

4. TITLE AND SUBTITLE

EVOLVING NEURAL NETWORK PATTERN CLASSIFIERS

5. FUNDING NUMBERS

PR: ZW67
PE: 0601152N
WU: DN303002

6. AUTHOR(S)

J. R. McDonnell, D. E. Waagen, and W. C. Page

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Chief of Naval Research
Arlington, VA 22217–5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. ABSTRACT *(Maximum 200 words)*

This work investigates the application of evolutionary programming for automatically configuring neural network architectures for pattern classification tasks. The evolutionary programming search procedure implements a parallel nonlinear regression technique and represents a powerful method for evaluating a multitude of neural network model hypotheses. The evolutionary programming search is augmented with the Solis & Wets random optimization method thereby maintaining the integrity of the stochastic search while taking into account empirical information about the response surface. A network architecture is proposed which is motivated by the structures generated in projection pursuit regression and the cascade-correlation learning architecture. Results are given for the 3-bit parity, normally distributed data, and the T-C classifier problems.

Published in *SPIE Conference on Neural and Stochastic Methods in Image and Signal Processing*, vol. 2032, July 1993.

14. SUBJECT TERMS

evolutionary programming
neural networks
signal detection

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE *(include Area Code)*	21c. OFFICE SYMBOL
J. R. McDonnell	(619) 553-5762	Code 731

SPIE Conf. on Neural and Stochastic Methods in Image and Signal Processing II, San Diego, CA, July, 1993.

Evolving neural network pattern classifiers

J.R. McDonnell, D.E. Waagen and W.C. Page
NCCOSC, RDT&E Div.
San Diego, CA

ABSTRACT

This work investigates the application of evolutionary programming for automatically configuring neural network architectures for pattern classification tasks. The evolutionary programming search procedure implements a parallel nonlinear regression technique and represents a powerful method for evaluating a multitude of neural network model hypotheses. The evolutionary programming search is augmented with the Solis & Wets random optimization method thereby maintaining the integrity of the stochastic search while taking into account empirical information about the response surface. A network architecture is proposed which is motivated by the structures generated in projection pursuit regression and the cascade-correlation learning architecture. Results are given for the 3-bit parity, normally distributed data, and the T-C classifier problems.

1. INTRODUCTION

Dynamic artificial neural networks (DANNs) represent an alternative training methodology which not only optimizes a weight set for a specified network architecture, but also allows the network architecture to be modified during the training process. The necessity for this type of training generally results from the trial-and-error process undertaken by the network designer on highly dimensioned data sets which lack obvious feature vectors. The usual objective of DANN training methodologies is to minimize an energy function that adequately describes the network topology as well as the mean sum-squared pattern error. This type of training approach results in parsimonious network structures with either a reduced number of redundant hyperplanes, minimized connectivity, or both. The benefits of the resulting networks include reduced throughput times for real-time signal processing applications and potentially better generalization capabilities by the avoidance of overfitting the training data.

The DANN training philosophy may be roughly broken down into two classes: (1) those that modify their connectivity and (2) those that modify the number of hidden units. Representative examples of the first class of DANN training algorithms include weight decay[1] and weight elimination[2]. Representative examples of the latter class of DANN training algorithms include the dynamic node creation (DNC) algorithm[3], an upgrade to the DNC algorithm by Hirose *et al.*[4] which also deletes units and the cascade-correlation (CC) learning architecture[5]. The CC training approach is unique in that it fixes the input-to-hidden unit weights and only modifies the weights of the output units. The idea of adding additional units to achieve better function approximations is similar to projection pursuit regression[6] techniques.

The use of evolutionary search methods is becoming prevalent as a network construction technique. Network architectures have been "evolved" during the training process using genetic algorithms[7], evolutionary strategies[8], and evolutionary programming[9]. Both the genetic algorithm and evolutionary strategy approaches incorporated the backpropagation algorithm for weight adjustment while the evolutionary programming approach implemented a hybrid stochastic search method.

The goal of this investigation is to determine the feasibility of an evolutionary search method, evolutionary programming, for the automatic design of a general feedforward neural network architecture. These architectures have three types of units (input, hidden, and output), as opposed to three types of layers. This distinction is made since each additional hidden unit may be connected to all of the previous units (both input and hidden) in the network. Citing the benefits of reduced connectivity given above, the resulting structures will not necessarily be fully connected in a CC sense. The next section briefly discusses the projection pursuit and CC classifier construction techniques. Note that other constructive approaches such as adaptive kernel estimators[10] are equally applicable for classifier determination. Finally, a hybrid learning architecture is proposed which, using evolutionary programming training methods, incorporates structural aspects of both the CC and projection pursuit architectures.

2. CLASSIFIER CONSTRUCTION MODELS

2.1 A connectionist representation of projection pursuit

Projection pursuit regression (PPR) structures are nonparametric models resulting from a successive refinement training process. Pattern classification using this regression technique has been demonstrated by Flick *et al.*[11]. PPR generates an approximation to $f(x)$ as a sum of empirically determined smooth functions g of linear combinations of the input vector as described by

$$\hat{f}_n(x) = \sum_{j=1}^{n} g_j(a_j^T x)$$

Training progresses by using a successive refinement concept to incrementally determine a ridge function $g_j(a_j^T x)$ with corresponding unit vector a_i which minimizes

$$E\left[r_m - g_m(a_m^T x)\right]^2$$

where r_j represents the current set of residuals. Flick *et al.*[11] point out that one problem with PPR is the inability to backfit the data by "readjusting the ridge functions used in earlier projections." A connectionist view can be taken of this regression technique with the resulting three-layer architecture illustrated in Fig. 1.

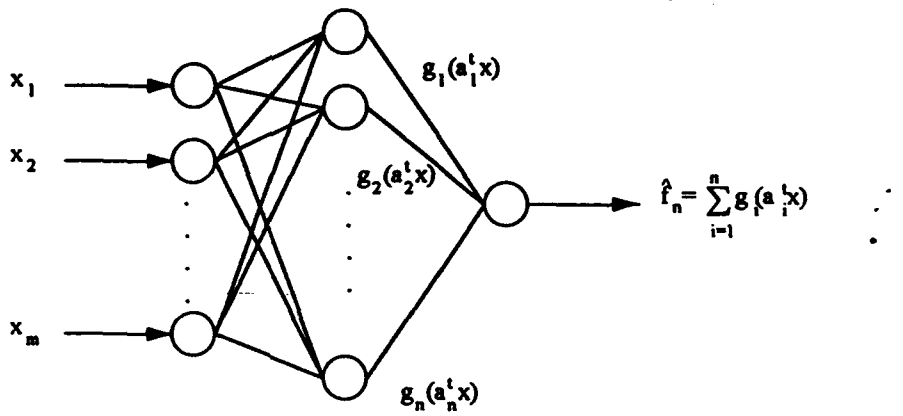


Fig. 1. A connectionist representation of the projection pursuit regression function.

2.2 The cascade-correlation algorithm

The successive refinement technique employed in PPR is similar to the construction method used in the CC learning architecture. The cascade-correlation learning architecture adds hidden units as necessary in an effort to minimize the residual errors. Significant differences between the CC and PPR algorithms are that the cascaded nodes can result in more complex nonlinear mappings whereas the PPR method selects an appropriate nonlinear mapping (and unit vector) to minimize the residuals at each generation. In the CC learning architecture a candidate pool of hidden units are individually trained to maximize the covariance between each units output and the residual output error over all output units. Once trained, the "best" hidden unit is incorporated into the network with fixed input weights with subsequent weight modifications occurring on the output units. The CC mapping is described by the network equations

$$\textit{output unit: } x_j = g\left(\sum_{i=1}^{m+n} w_{ij} x_i\right) \quad \textit{for} \quad j > m+n \; ; \qquad \textit{hidden unit: } x_j = g\left(\sum_{i=m}^{j-1} w_{ij} x_i\right) \quad \textit{for} \quad j > m$$

where wi_j corresponds to the weight matrix and m and n represent the size of the input vector and number of hidden units, respectively. For feedforward architectures, the weight matrix is nearly upper triangular. The CC architecture is shown in Fig. 2.

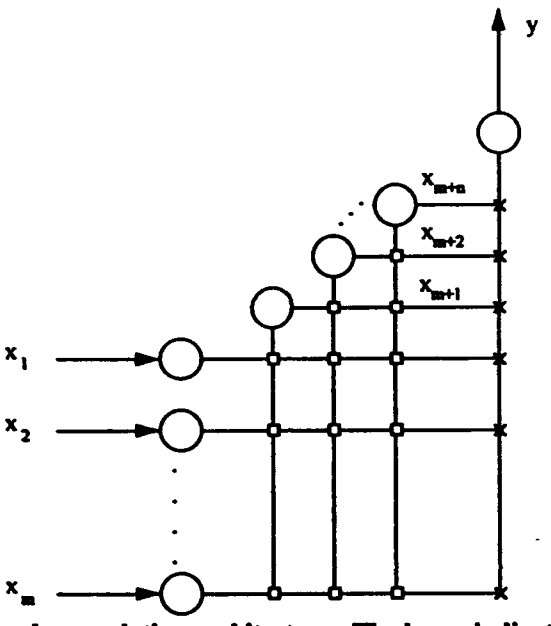


Fig. 2. The cascade-correlation architecture. The boxes indicate weights which are frozen, only the output weights are modified.

The difference between the CC and PPR function approximation technique previously discussed becomes more evident if the output unit equation is rewritten with separate summations for the input and hidden units and expanded accordingly

$$y_j = g\left(\sum_{i=1}^{m} w_{ij} x_i + \sum_{i=m+1}^{m+n} w_{ij} x_i\right)$$

$$y_j = g\left(\sum_{i=1}^{m} w_{ij} x_i + w_{m+1,j} x_{m+1} + \cdots + w_{m+n,j} x_{m+n}\right)$$

$$y_j = g\left(\sum_{i=1}^{m} w_{ij} x_i + w_{m+1,j}\, g\left(\sum_{i=1}^{m} w_{i,m+1} x_i\right) + \cdots + w_{m+n,j}\, g\left(\sum_{i=1}^{m+n-1} w_{i,m+n} x_i\right)\right)$$

where the subscript on g has been dropped signifying the activation functions are the same (this is not always the case). Instead of the linear combination of nonlinear mappings of projections of the input vector x as generated using the PPR algorithm, the CC architecture yields a nonlinear mapping of a weighted linear combination of nonlinear mappings of projections of the input vector x. As the number of hidden units n increases, more complex nonlinear manifolds result. In comparison to the PPR mapping, this capability may yield better pattern classification results on highly non-convex data sets. This deficiency of PPR with respect to its ability to achieve highly nonlinear mappings is also pointed out by Huber[12] who states "that PP is poorly suited to deal with highly nonlinear structures."

3. EVOLUTIONARY PROGRAMMING

Evolutionary programming (EP) provides a powerful framework for simultaneously evaluating neural network models and their parameterizations. Like PPR methods, the EP search strategy can be computationally demanding. EP is a systematic, multi-agent, stochastic search technique proposed by Fogel *et al.*[13]. EP has been used to generate finite state machines[13], auto-regressive moving average (ARMA) models[14], probability density mixture models[15], and recurrent perceptrons[16].

The EP paradigm can be described by the following algorithm[14]

1. *Form an initial population* $P = [x_0 x_1 x_2 \ldots x_{2N-1}]$ *of size 2N by randomly initializing each n-dimensional solution vector* x_i. *A user specified search domain* $x_i \in [x_{min}, x_{max}]^n$ *may be imposed.*
2. *Assign a cost to each element* x_i *in the population based on the associated objective function* $J_i = \Phi(x_i)$ s.t. $\Phi: R^n \rightarrow R$
3. *Reorder the population in descending order based on the number of wins generated from a stochastic competition process. Wins are generated by randomly selecting other members in the population* x_j *and incrementing the win counter* w_i *if* $J_i < J_j$.
4. *Generate offspring* $(x_N \ldots x_{2N-1})$ *from the N highest ranked elements* $(x_0 \ldots x_{N-1})$ *in the population by modifying each element* $x_{ij} \in x_i$ *with a random perturbation* $\delta x_{ij} \sim N(0, S_{f,ij} \cdot J_i + \beta_{ij})$ *such that* $x_{i+Nj} = x_{ij} + \delta x_{ij}$
5. *Loop to step 2.*

The Bohachevsky function $f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$ is proposed as a response surface to demonstrate the benefits of EP as a global optimization strategy. The transcendental terms generate many local minima within the interval $x \in [-1,1]^2$ while the quadratic terms dominate the surface structure outside of this interval. A unique global minimum exists at $x=(0,0)$. A trajectory of the best population member at each generation during a search on the Bohachevsky surface is superimposed on the Bohachevsky contours as shown in Fig. 4. Since the search is stochastic, it is expected that this trajectory will vary for every trial.

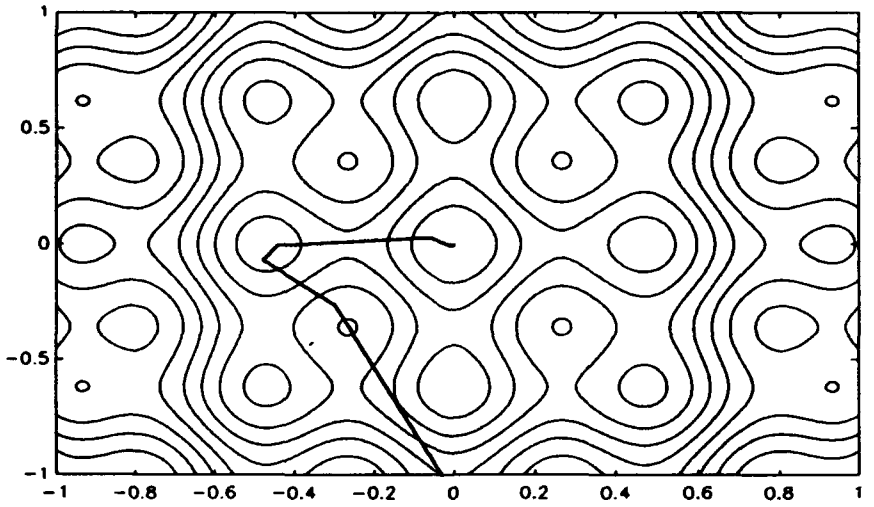


Fig. 4. Trajectory of the best point in the population during the evolutionary search process on the Bohachevsky surface.

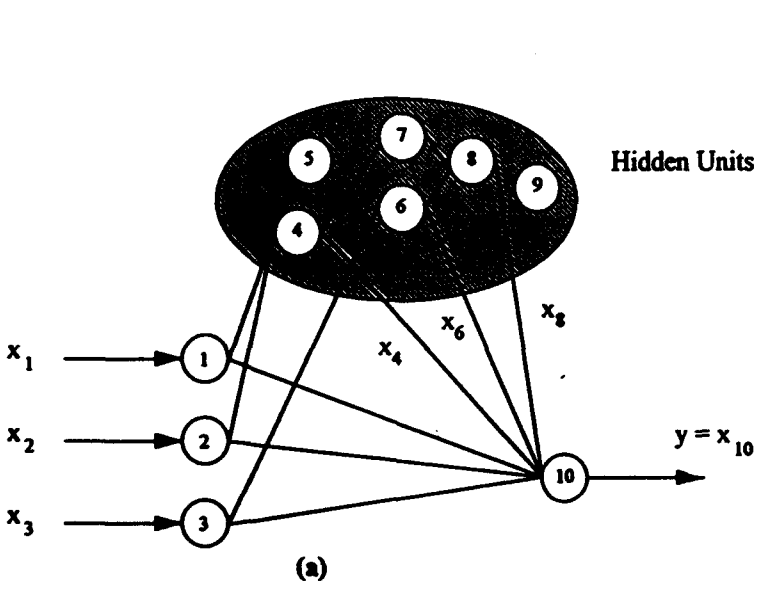
2.3 The neural network classifier model

A feedforward neural network may be considered as a functional mapping $f:X \rightarrow Y$ where $X \in R^m$, $Y \in R^n$ subject to a topology *T(N,C)* as defined by the neuron inter-connectivity C over the number of available neurons N. Similar to the PPR algorithm, the mapping network may even contain variable types of activation functions g so that $g \in G$ where G is the set of possible activation functions. Motivated by the PPR and CC structures, as well as the better generalization capabilities of parsimonious structures, a general structure shown such as that shown in Fig. 3(a) is proposed as a pattern classifier. It is the intention of this work to generate such architectures using the evolutionary programming search strategy.

Differences between the proposed architecture and the CC structure include the ability to modify all connection weights throughout the learning process. This is different than minimizing the residuals with each additional hidden units as accomplished in the CC learning architecture. To achieve less than full connectivity, a connectivity array C is specified as shown in Fig. 3(b) where the row index corresponds to "from" units and the each column index corresponds to "to" units (*i.e., C[from][to]*). If each connectivity-weight product is combined as $\alpha_{ij} = c_{ij} \cdot w_{ij}$, then the nonlinear mapping is the similar to the CC mapping (assuming $c_{ij}=1$) and can be described as

$$y_j = g_j\left(\sum_{i=1}^{m} \alpha_{ij} x_i + \alpha_{m+1,j} g_{m+1}\left(\sum_{i=1}^{m} \alpha_{i,m+1} x_i\right) + \cdots + \alpha_{m+n,j} g_{m+n}\left(\sum_{i=1}^{m+n-1} \alpha_{i,m+n} x_i\right)\right)$$

where variable activation functions have been incorporated. While activation functions could easily be incorporated as an additional evolutionary search parameter, the results presented in this work maintained the same activation functions on the output and hidden units since this is a preliminary investigation. Training must not only determine the network weights, but also the neuron inter-connectivity. This problem requires an approach which address NP-hard optimization problems. The technique employed in this investigation is the evolutionary programming method. It may be argued that desired portions of the network are needlessly altered by modifying all of the free parameters during the search process. As will be seen in Section 4, the evolutionary search is conducted in a manner that does not simultaneously effect all of the network parameters. It is expected that surviving members of the population will retain beneficial structure and parameters.



$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b)

Fig. 3. (a) A general feedforward network architecture and (b) its associated connectivity matrix.

It is interesting to note that evolutionary search strategies are generally robust with respect to a broad class of problems. Nontrivial constraints can be incorporated into the objective function in an effort to take advantage of *a priori* knowledge about the problem domain. It should also be noted that evolutionary optimization strategies tend to be slower than more deterministic optimization approaches. However, their "time complexity quite often grows in a linear manner together with the problem size" [17].

4. EVOLVING ANN PATTERN CLASSIFIERS

As previously stated, EP provides a powerful relaxation search. To improve search efficiency, algorithm 1 of Solis&Wets[18] has been embedded in the search process in parallel with the offspring method normally incorporated in EP. A connection modification mechanism is implemented by changing the state of a randomly selected synapse. For example, if *C[i][j] = 1* then the bit is flipped to 0. Likewise, if *C[i][j] = 0* then the bit is flipped to 1. The structure modification procedure must be employed with caution. Frequent structural modifications, say every generation, will cause the network to evolve with a high MSE and no connections. This happens when learning is slow to occur with respect to the frequency of structural modifications. As a result, the connection matrix is modified only every K generations (K=*5-10* for this work). These changes are manifested in the evolutionary search strategy by replacing step 4 in the EP algorithm above with

4. *Generate offspring*
   - i. *Modify the N highest ranked elements (x_0 .... x_{N-1}) using the Solis & Wets algorithm 1*
   - ii. *Perturb the weight set using the EP scheme discussed above and modify the connectivity structure of (x_N .... x_{2N-1}) by flipping a randomly selected bit in the connectivity matrix every K generations.*

By replacing the parent networks with offspring generated using the Solis & Wets algorithm, it is guaranteed that the objective function will be decreasing, that is $\Delta J_i = J_i(k+1) - J_i(k) \le 0$. New structures are generated in the offspring with "good" structures being propagated at the parent level. A relatively small population size (N=*10* with single offspring) is used in this investigation. A single generation of the evolutionary search procedure is shown in Fig. 5.

The cost function employed in this study is a heuristic based on Akaike's[19] information criterion (AIC). This information criterion address model order by incorporating the number of free parameters to be determined for a particular model selection along with the maximum likelihood estimate. Other information criterion such as the minimum description length (MDL) could also have been used for the objective function.

The AIC for an autoregressive moving-average (ARMA) model with AR order p and MA order q as described by

$$AIC(p,q) = P\log(\hat{\sigma}_e^2) + 2(p+q)$$

has been modified for use as the objective function so that

$$AIC(N_c) = P\log(\hat{\sigma}_e^2) + 2N_c$$

where P is the effective number of observations or patterns and N_c is the number of connections. Lower AIC values indicate better models. Thus, the goal of the evolutionary search process is to minimize $AIC(N_c)$ by searching over the weight and connectivity spaces. The maximum likelihood estimate of the variance is determined in the usual way[20]

$$\hat{\sigma}_e^2 = \frac{1}{P}\sum_p \sum_i \left(t_{p,i} - o_{p,i}\right)^2$$

where $t_{p,i}$ is the desired target and $o_{p,i}$ is actual output of neuron i for input pattern p. For the present study, it is suspected that this information criterion insufficiently addresses the number of free parameters utilized in the network. Nevertheless, it is employed without regard for the number of hidden units implemented in the network.

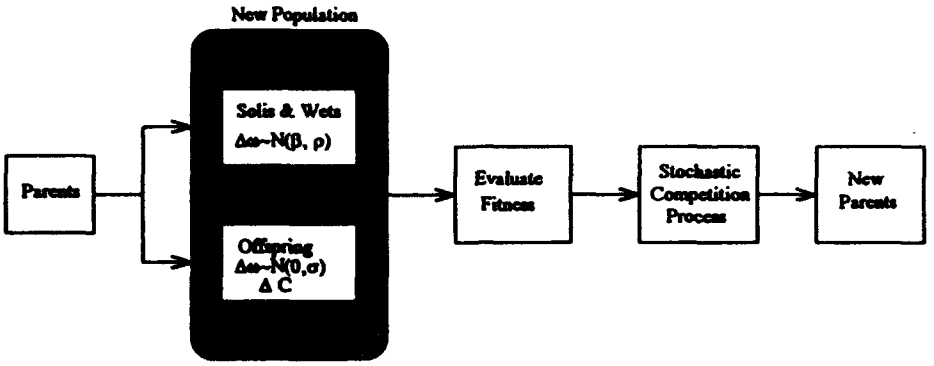


Fig. 5. One generation of the hybrid multi-agent stochastic search. This method is a combination of both EP and the Solis&Wets technique.

5. RESULTS

5.1 The parity problem

The parity problem serves as a popular benchmark since the mapping is not linearly separable. Tests were conducted for the three-bit parity problem which consists of eight exemplars. Sigmoidal activation functions were incorporated in the network. Ten parent networks were employed, each having a single offspring. The networks were initialized to be fully connected at the start of the run to five hidden units. An example evolutionary optimization run is shown in Fig. 6 which shows the number of connections and MSE of the network with the least cost at each generation. Even after the MSE reaches an acceptable level the optimization procedure continues to reduce the number of connections. It is interesting to observe which part of the search yields the best network at each generation. For the run shown in Fig. 6, Fig. 7 gives the index of the network with the lowest cost. Indices below 10 indicate the Solis & Wets algorithm generates better nets and indices equal to or greater than 10 illustrate that the EP weight perturbation/connection modification strategy generates lower cost nets. It should be noted that the network generated in this run is essentially a two-layer network with shortcut connections from the input units to the hidden units as shown in Fig. 8. Recall that a traditional fully-connected architecture with a single hidden layer would result in 16 connections whereas the network shown in Fig. 8 has 13.

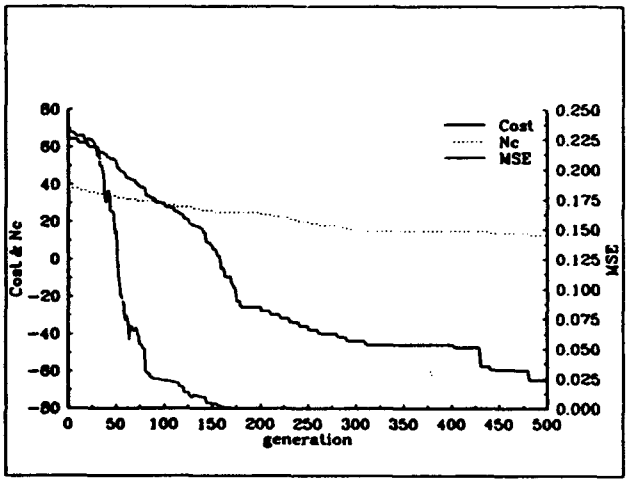


Fig. 6. The evolutionary optimization procedure applied to the 3bit parity problem. The network with the lowest cost is shown at each generation.

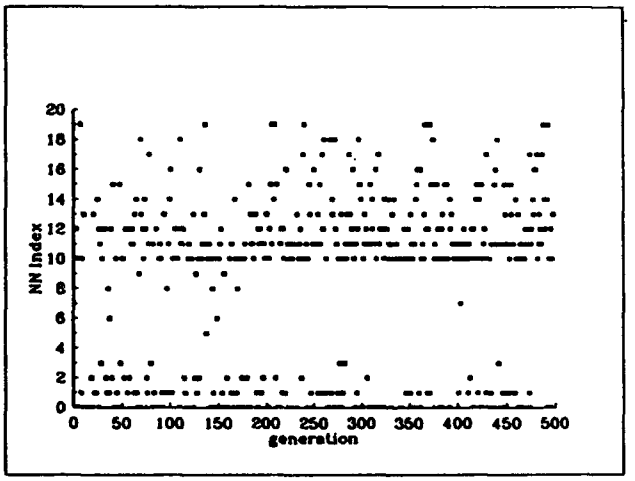


Fig. 7. The NN index indicates the network rank in the population. The rank of the network with the lowest cost is shown at each generation.

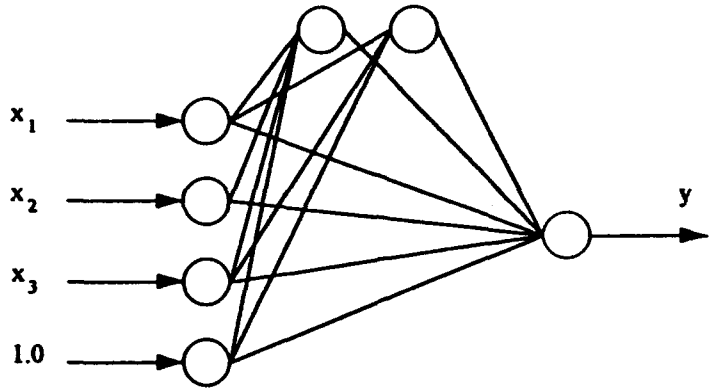


Fig. 8. Resulting network configuration for 3bit parity problem.

5.2 A two-class Gaussian problem

Let the function $f(x,y)$ be jointly normal as denoted by $N(\mu_x,\mu_y,\sigma_x,\sigma_y)$. A small sample of 50 observations is taken from class 1 as defined by $N(-1.5,0,1,1)$ and class 2 as defined by $N(1.5,0,1,1)$, respectively. Using sigmoidal activation functions, a network was evolved to distinguish between the two classes given a single (x,y) observation. Again, this problem is of academic interest due to its nonlinear separability requirement. A modification was made in the evolutionary learning procedure for this mapping. Since the Solis & Wets method is a powerful random optimization technique in its own right, it was used to generate offspring from (Solis & Wets) modified parent networks in lieu of generating offspring in the traditional EP fashion. The EP framework (*i.e.* competitive annealing) was still used to retain good network structures.

The cost and MSE of the best network at each generation is shown in Fig. 9. Fig. 10 shows the index of the best network at every generation. Using the scheme described above, the offspring networks will always be equivalent to or better than the parent networks unless the connectivity structure is modified. The network was initially fully connected to 10 hidden units (88 connections). After 5000 generations only 38 connections remained with a MSE=0.0003. The resulting configuration is shown in Fig 11. Fig. 12 shows the limited number of samples from each class superimposed on the contour plot. Duda and Hart[21] give more determinstic methods for formulating discriminants if normal densities are assumed.

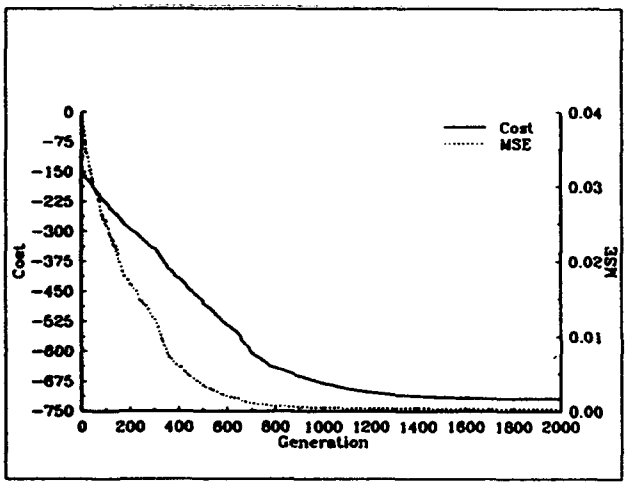


Fig. 9. Evolutionary optimization for a two-class Gaussian data set.

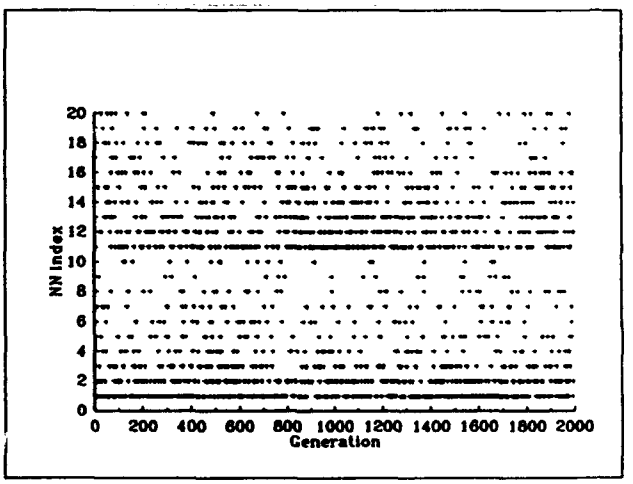


Fig. 10. The network with the lowest cost at each generation. Networks 1-10 and 11-20 correspond to the parent and offspring networks, respectively.

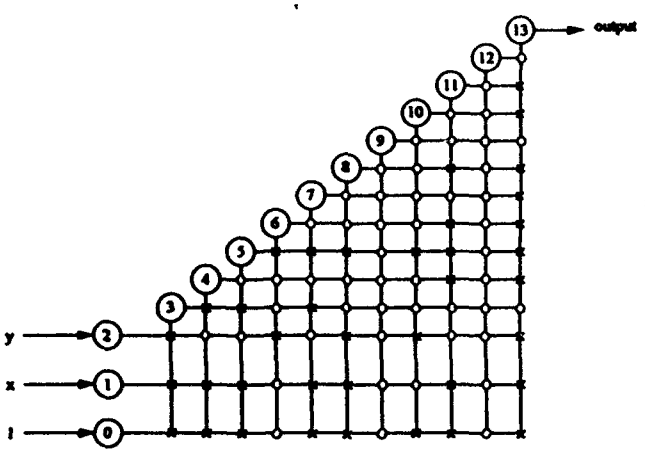


Fig. 11. The resulting network for the two-class Gaussian problem after 5000 generations. The x's indicate connections whereas the o's indicate links which are not connected. Note units 9 and 12 have not been incorporated.

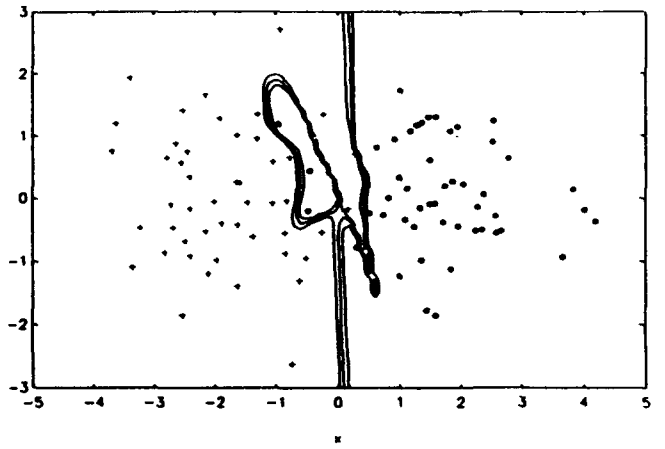


Fig. 12. The two classes of data superimposed on a contour plot of the decision surface. The '+' class has an output value of 1 and the 'o' class has an output value of 0.

5.3 The T-C classifier

The final set of computer experiments investigates the classification of binary T-C images which have different scaling and rotation as shown in Fig. 13. The eight 25 bit "T" patterns were designated as a separate class from the eight 25 bit 'C' patterns. Starting with a population of fully-connected networks with 10 hidden units, it generally took less than 500 generations to evolve a network which distinguishes between the given "T" and "C" patterns. Fig. 14 shows the evolutionary optimization process and Fig. 15 shows the resulting network. Since the center pixel is always 1, it is interesting to note that the input from this pixel was not disconnected as it does not provide any discriminatory information between the two patterns. It is speculated that this probably would occur with an increased number of learning generations. For the network shown in Fig. 14, the number of connections was reduced roughly 11% (from 341 to 302). However the MSE appears acceptable within the short number of generations. Most of the trials generated an 11-13% reduction in 500 generations.

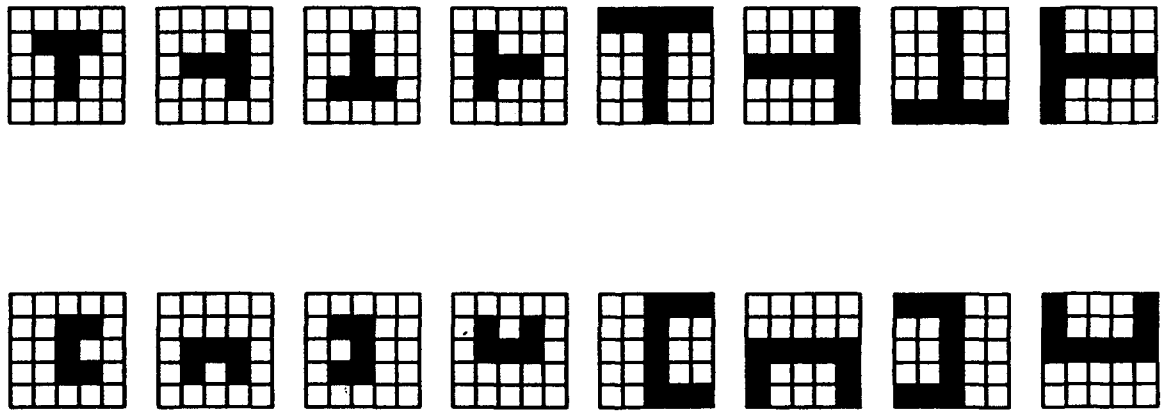


Fig. 13. Binary T-C patterns rotated and scaled on a 5x5 grid.

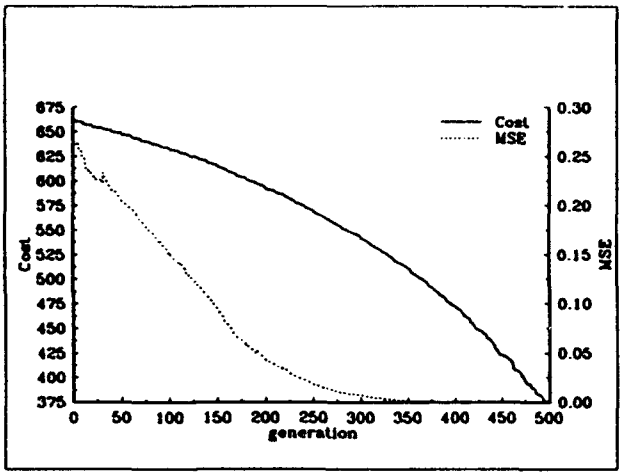


Fig. 14. Evolutionary optimization for the T-C patterns shown in Fig. 13. Training was arbitrarily stopped at 500 generations. It appears that the cost function is still being minimized by disconnecting neurons.

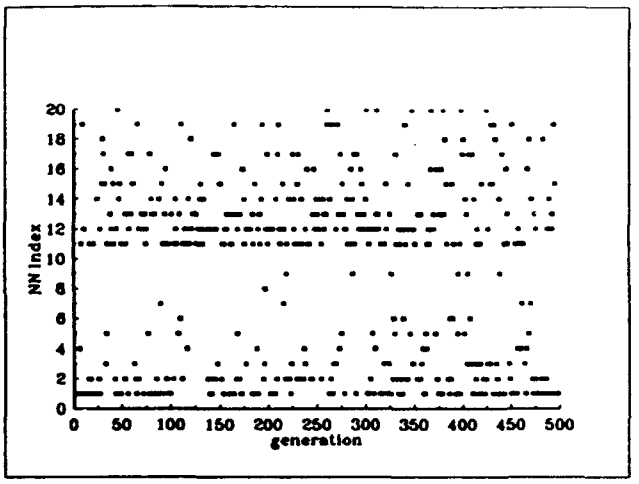


Fig. 15. The ranking of the best network in the population at each generation. As in the previous section a dual Solis & Wets approach was used to both replace the parents and generate offspring.

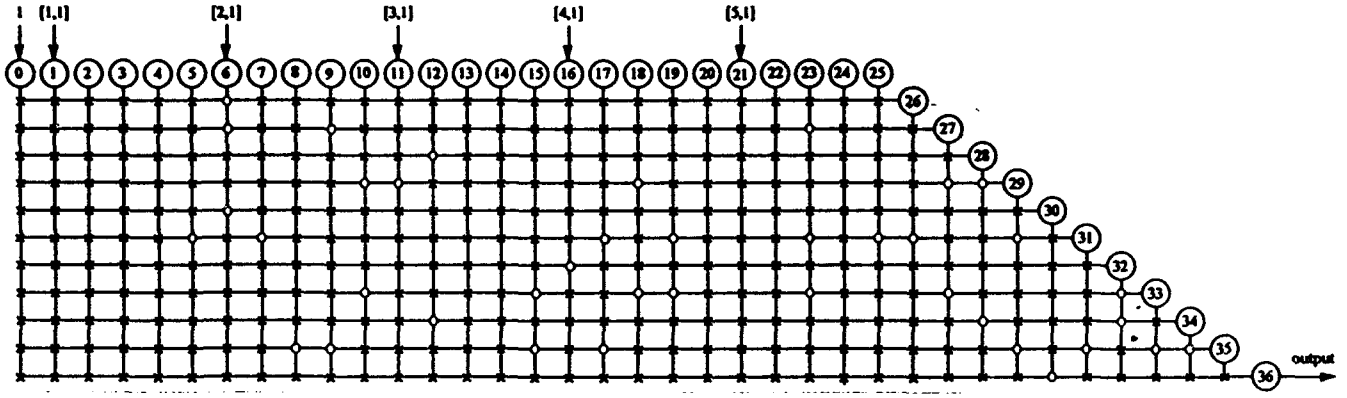


Fig. 16. The evolved network for solving the T-C pattern problem given in Fig. 13. It is very likely that additional link would have been disconnected given more learning generations.

6. CONCLUSION

The network designer imposes certain constraints in selecting a network architecture before training. These constraints are manifested in the topology which describes the inter-neuron connections, the number of neurons and the activation function. Once the network architecture is arbitrarily chosen, a weight set is found which optimizes the model for the desired mapping. Constructive techniques such as projection pursuit regression and cascade-correlation architectures serve as a means to relax the constraints imposed by the designer. The global optimization capabilities of evolutionary search methods can be used to generate subsets of cascade-correlation style architectures by simultaneously searching over weight and neuron connectivity spaces. Since these techniques are stochastic and global in nature, once good solutions are found they may be optimized using local methods.

The evolutionary optimization approach outlined in this paper is extremely versatile. Although static activation functions were employed for each neuron, changing the activation function did not require modification of the optimization code. Spurious connections normally generated by evolutionary construction of networks were not observed. Reasonable results were consistently found for the types of patterns classified in this work even though a small number of parent networks where used. Additional work will ascertain the better generalization capabilities, if any, achieved using

parsimonious structures generated by this approach. Further work will also apply this technique to more difficult mappings such as the two-spiral classification problem[5] as well as classification problems of Naval interest.

7. ACKNOWLEDGMENT

The authors wish to thank Dr. Al Gordon, program director for NRaD internal research projects, for his support of this work.

8. REFERENCES

1. J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

2. A.S. Weigend, B.A. Huberman, and D.E. Rumelhart, "Predicting sunspots and exchange rates," in *Nonlinear Modeling and Forecasting*, M. Casdagli and S. Eubank (ed.), Addison-Wesley, 1992.

3. T. Ash, "Dynamic node creation in backpropagation networks," Technical Report 8901, Institute for Cognitive Science, University of California, San Diego, 1989.

4. Y. Hirose, K. Yamashita, and S. Hijiya, "Backpropagation algorithm which varies the number of hidden units," *Neural Networks*, Vol. 4, 1991.

5. S.E. Fahlman and C.L. Lebiere, "The cascade-correlation learning architecture," Technical Report CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, PA, 1990. Also in Touretzky (ed.): *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, 1990.

6. J.H. Friedman and W. Stuetzle, "Projection pursuit regression," *Journal of Am. Stat. Ass.*, Vol. 76 ,No. 376, pp. 817-823, Dec., 1981.

7. G.F. Miller, P.M. Todd, and S.U. Hegde, "Designing neural networks using genetic algorithms," *Proc. of the Third Int. Conf. on Genetic Algorithms*, San Mateo, CA, 1989.

8. H-M. Voight, J. Born, and I. Santibanez-Koref, "Evolutionary structuring of artificial neural networks," Technical Report TR-02-03, Technical University Berlin, Berlin, 1992.

9. J.R. McDonnell and D. Waagen, "Neural network structure design by evolutionary programming," *Second Annual Conf. on Evolutionary Programming*, San Diego, 1993.

10. C.E. Priebe and D.J. Marchette, "Adaptive mixtures: recursive nonparametric pattern recognition," *Pattern Recognition*, Vol. 24, pp. 1197-1209, 1991.

11. T.E. Flick, L.K. Jones, R.G. Priest, and C. Herman, "Pattern classification using projection pursuit," *Pattern Recognition*, Vol. 23, No. 12, pp. 1367-1376, 1990.

12. P.J. Huber, "Projection pursuit," The Annals of Statistics, Vol. 13, No. 2, pp. 435-475, 1985.

13. L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons, 1966.

14. D. B. Fogel, *System Identification though Simulated Evolution: A Machine Learning Approach to Modeling*, Ginn Press, Needham, MA, 1991.

15. D. Waagen and J.R. McDonnell, "Probability density model and parameter estimation using evolutionary programming," *Second Annual Conf. on Evolutionary Programming*, San Diego, 1993.

16. J.R. McDonnell and D. Waagen, "Evolving recurrent perceptrons for time-series prediction," unpublished.

17. Z. Michalewicz, *Genetic Algorithms+Data Structures = Evolution Programs*, Springer-Verlag, 1992.

18. F.J. Solis and R.J-B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, Vol. 6, No. 1, pp. 19-30, 1981.

19. H. Akaike, "A new look at the statistical model identification," IEEE Trans. on Automatic Control, Vol. 19, No. 6, pp. 716-723, 1974.

20. M.B. Priestley, *Nonlinear and Nonstationary Time Series*, Academic Press, 1988.

21. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.